LSDIS

Large Scale Distributed Information Systems

University of Georgia
Computer Science Department

Peer-to-Peer Discovery of Semantic Associations

Matthew Perry, Maciej Janik, Cartic Ramakrishnan,
Conrad Ibanez, Budak Arpinar, Amit Sheth

2nd International Workshop on Peer-to-Peer Knowledge Management,
San Diego, California, July 17, 2005

Semantic Discovery[1]

From …..

Finding things

To …..

Finding out about things

Relationships!

1. http://lsdis.cs.uga.edu/semdis

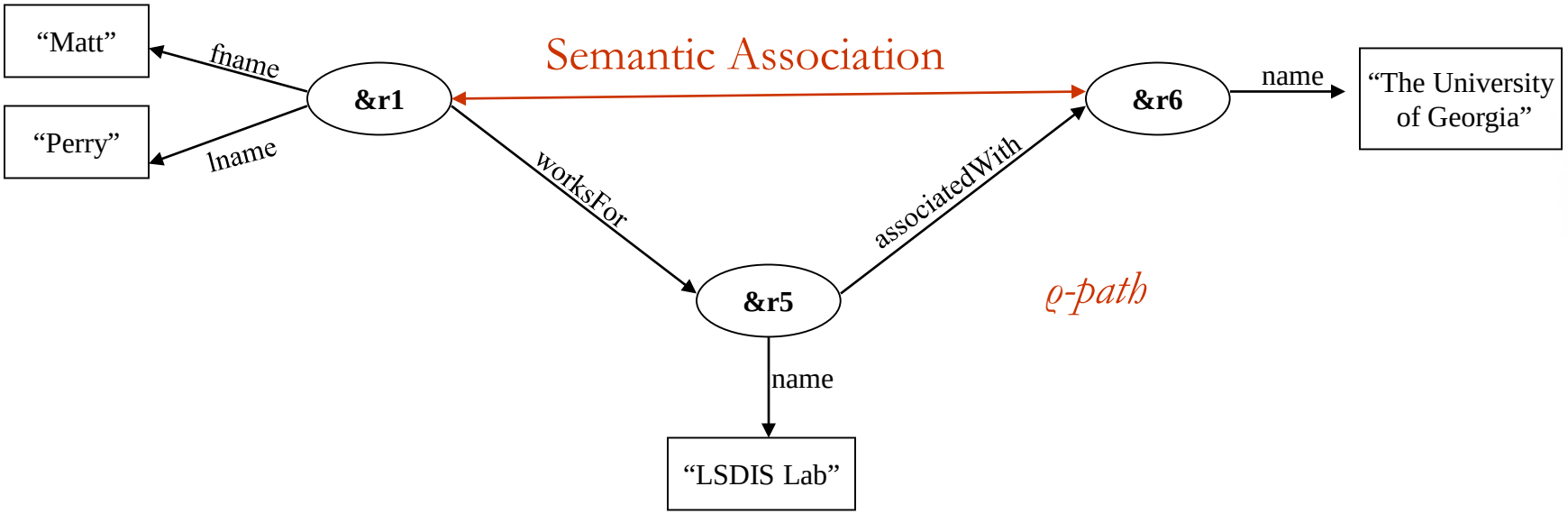
Semantic Associations

- Relationship-centric nature of Semantic Web data models
- We can ask questions about the relationships between objects
- How is entity *A* related to entity *B*?
- Applications
  - National Security – Insider Threat[1]
  - Improved Searching – Bio Patent Miner[2]

1. B. Aleman-Meza, P. Burns, M. Eavenson, D. Palaniswami, A. Sheth, An Ontological Approach to the Document Access Problem of Insider Threat, Proceedings of the IEEE Intl. Conference on Intelligence and Security Informatics (ISI-2005), May 19-20, 2005
2. Sougata Mukherjea, Bhuvan Bamba, BioPatentMiner: An Information Retrieval System for BioMedical Patents, VLDB 2004.

Semantic Associations

Define a set of operators ρ for querying complex relationships between entities (Semantic Associations)[1]



1. Adapted From: Kemafor Anyanwu, and Amit Sheth, ρ-Queries: Enabling Querying for Semantic Associations on the Semantic Web, The Twelfth International World Wide Web Conference, Budapest, Hungary, pp. 690-699.

Uniqueness of Semantic Association Queries

- Simple query specification (only the two endpoints)
- Doesn't require extensive knowledge of schema

ρ-path (*A*, *B*)

Difficult to express with existing Query Languages

```
SELECT ?startURI, ?property_1, ?endURI
FROM (?startURI ?property_1 ?endURI)

SELECT ?startURI, ?property_1, ?endURI
FROM (?endURI ?property_1 ?start)

SELECT ?startURI, ?property_1, ?x, ?property_2, ?endURI
FROM (?startURI ?property_1 ?x)(?x ?property_2 ?endURI)
WHERE ?startURI ne ?x && ?endURI ne ?x

SELECT ?startURI, ?property_1, ?x, ?property_2, ?endURI
FROM (?startURI ?property_1 ?x)(?endURI ?property_2 ?x)
WHERE ?startURI ne ?x && ?endURI ne ?x

SELECT ?startURI, ?property_1, ?x, ?property_2, ?endURI
FROM (?x ?property_1 ?startURI)(?x ?property_2 ?endURI)
WHERE ?startURI ne ?x && ?endURI ne ?x

SELECT ?startURI, ?property_1, ?x, ?property_2, ?endURI
FROM (?x ?property_1 ?startURI)(?endURI ?property_2 ?x)
WHERE ?startURI ne ?x && ?endURI ne ?x
```

RDQL: Find paths of length at most 2 from startURI to endURI

Why Semantic Associations in P2P?

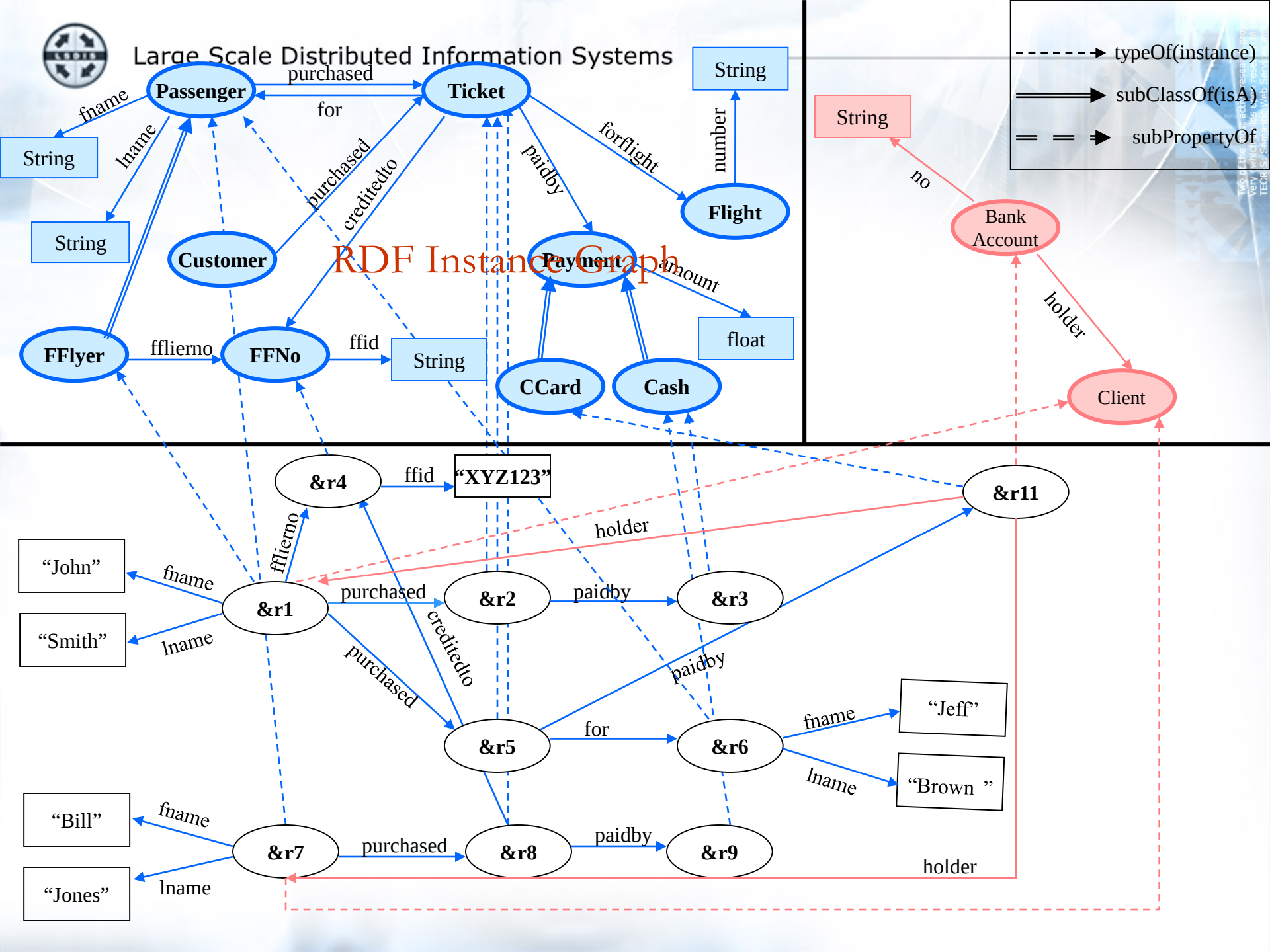
- Data on the web by its nature is distributed
- Knowledge will be stored in multiple stores and multiple ontologies
- Search for semantic paths will have to include many knowledge sources
- In the spirit of the Semantic Web (collaborative knowledge discovery)

Contributions

- Super-Peer Architecture for Querying Semantic Associations
- Knowledgebase Borders and Distances between Borders
- Query Planning Algorithm based on Knowledgebase Borders and Distances

Assumptions

- Pair-wise mapping of resources between peers (solution to Entity Disambiguation / Reference Reconciliation problem)
- We use URIs to solve Entity Disambiguation problem
- Main focus is Query Planning over P2P network
- Not concerned with fault tolerance, details of network formation, etc. at this point

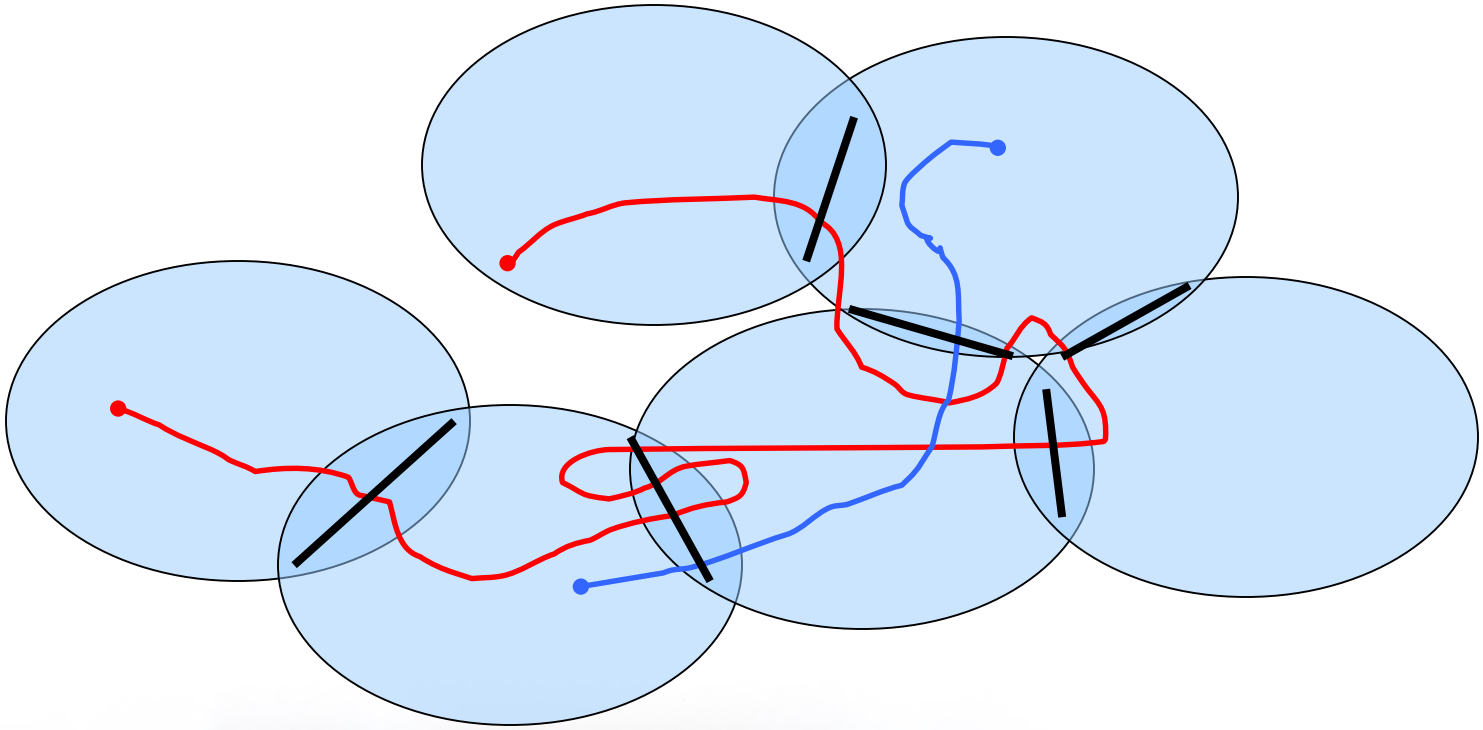

Large Scale Distributed Information Systems
RDF Instance Graph
typeOf(instance)
subClassOf(isA)
subPropertyOf
Passenger
Ticket
Flight
Customer
Payment
FFlyer
FFNo
CCard
Cash
Bank Account
Client
String
float
purchased
for
fname
lname
number
forflight
paidby
creditedto
fflierno
ffid
amount
no
holder
&r1
&r2
&r3
&r4
&r5
&r6
&r7
&r8
&r9
&r11
"XYZ123"
"John"
"Smith"
"Jeff"
"Brown "
"Bill"
"Jones"

ρ-*path* Problem (*k*-hop limited)

- Given:
  - An RDF instance graph *G*, vertices *a* and *b* in *G*, an integer *k*
- Find:
  - All simple, undirected paths *p*, with length less than or equal to *k*, which connect *a* and *b*

Distributed ρ-path problem: Find all paths from a *start* node to an *end* node over the distributed RDF graphs

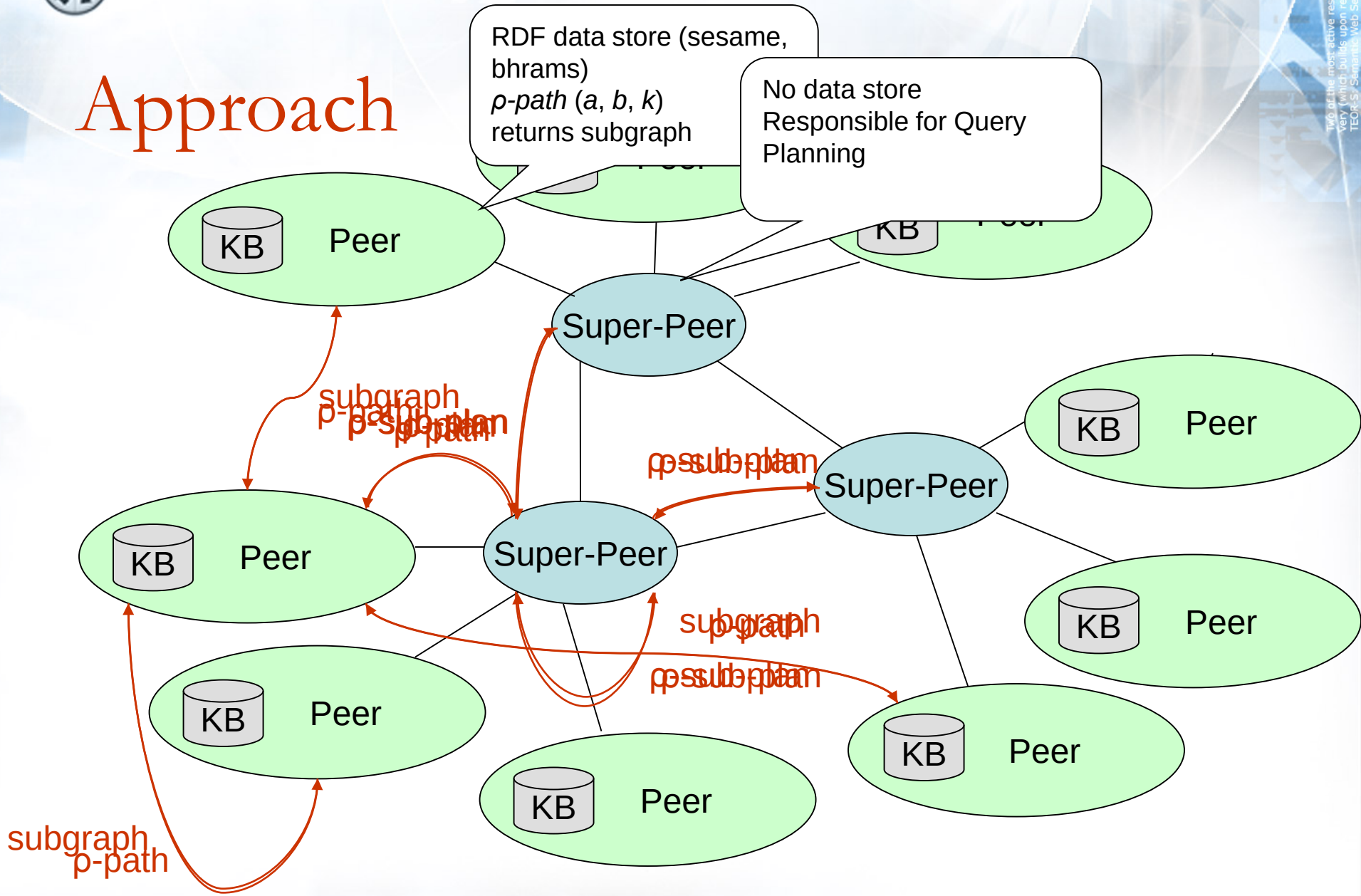
Knowledge bases - ontologies



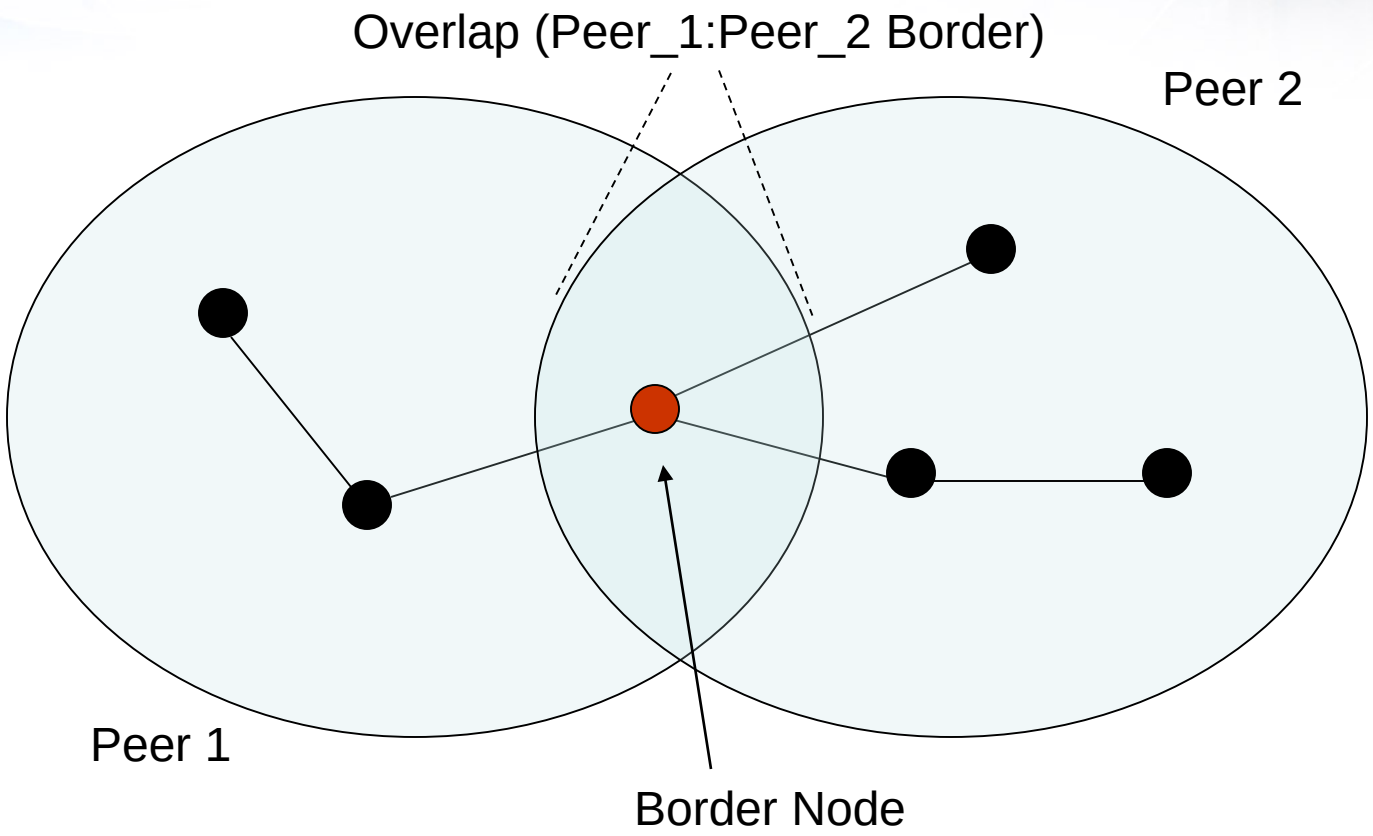
What do we need?

- Efficiently explore node neighborhoods
- When to stop a search in one peer and continue it in another
- Determine the search distance in each peer
- Determine which peers to include in the search

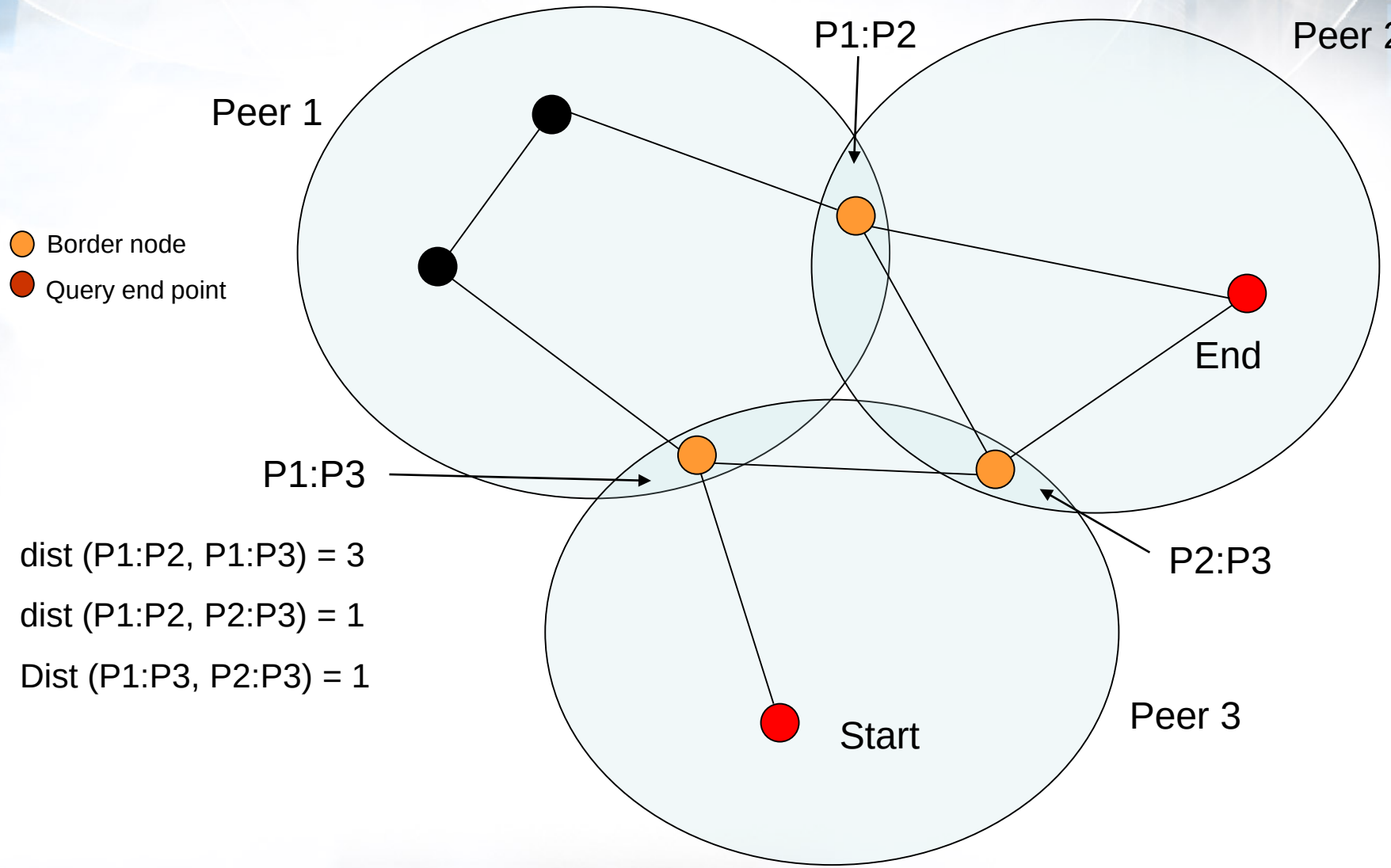
Approach


RDF data store (sesame, bhrams)
p-path (*a*, *b*, *k*) returns subgraph
No data store
Responsible for Query Planning
KB
Peer
Super-Peer
subgraph
p-path

Knowledgebase Borders



Distance Between Borders



dist (P1:P2, P1:P3) = 3

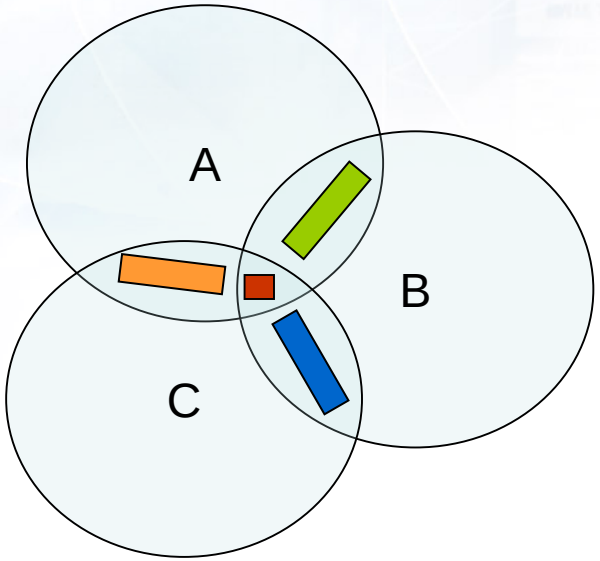
dist (P1:P2, P2:P3) = 1

Dist (P1:P3, P2:P3) = 1

Query Planning Graph

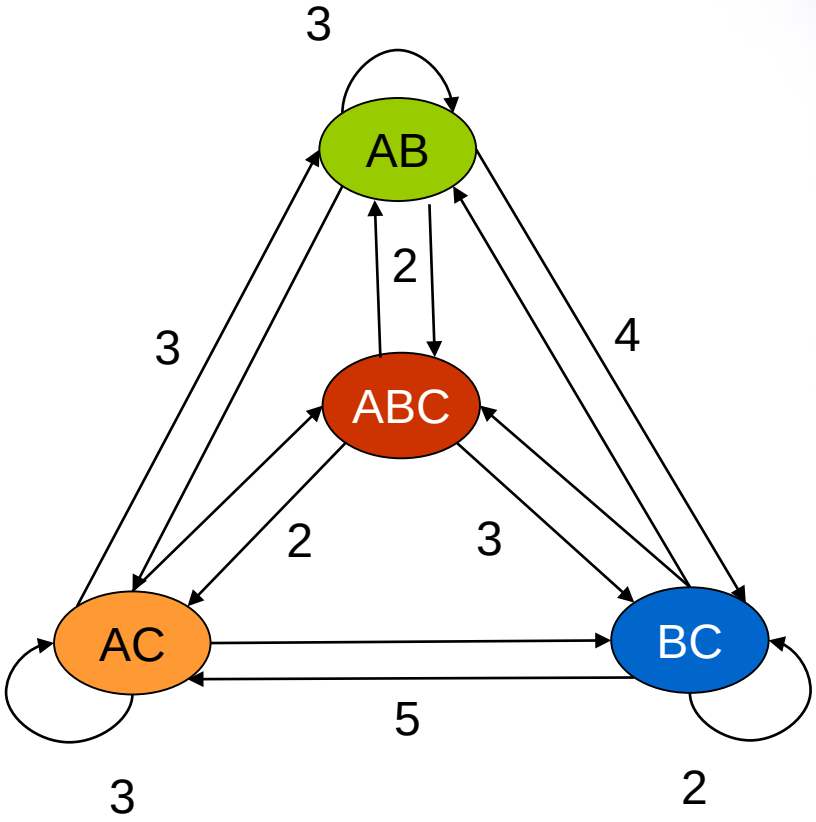
- Directed Graph
- Node for each distinct border
- For each pair of connected borders, create 2 edges (one in each direction)
- Weight is the minimum of the minimum distances (reported by peers)
  - For example you can get from A:B to A:B:C through either A or B

Query Planning Graph

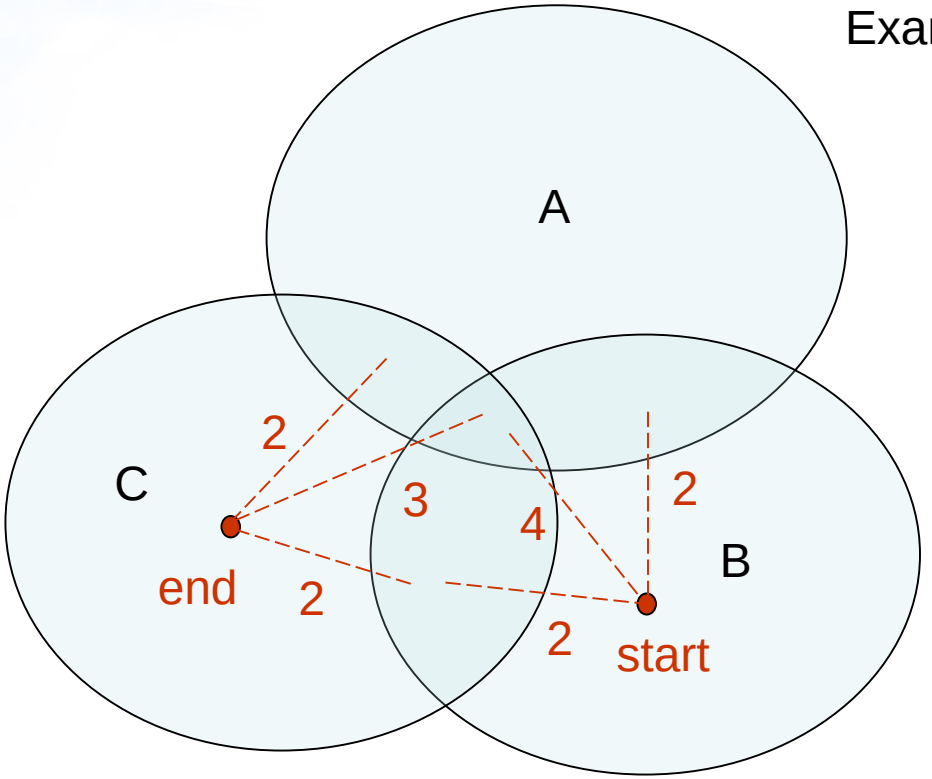


Minimum Distances
dist (AB, BC) = 4
dist (AB, AC) = 3
dist (AB, ABC) = 2
dist (BC, AC) = 5
dist (BC, ABC) = 3
dist (AC, ABC) = 2
dist (AB, BB) = 3
dist (AC, AC) = 3
dist (BC, BC) = 2
dist (ABC, ABC) = ∞

Borders
AB
AC
BC
ABC



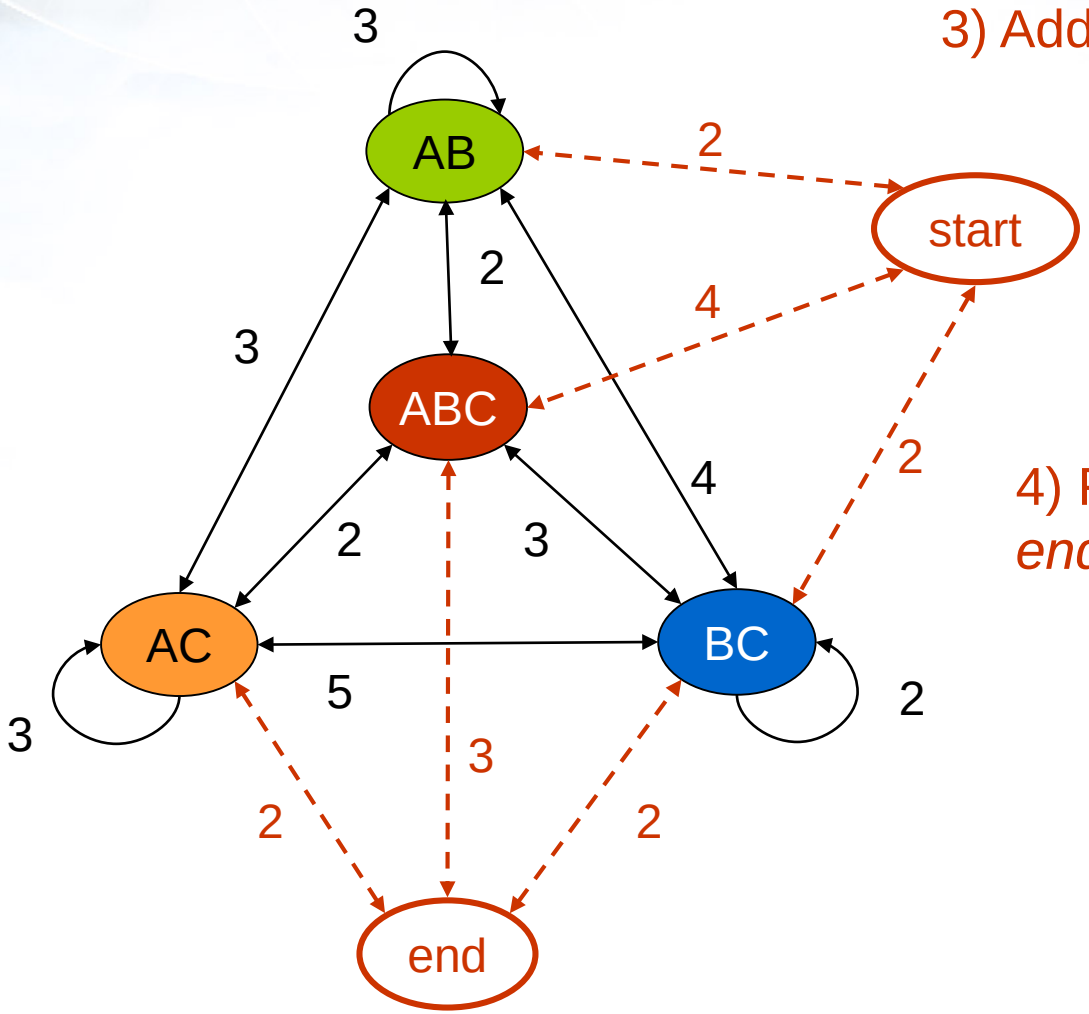
Using the Query Planning Graph



Example Query: r-path (start, end, 10)

1) Find Start and End Points

2) Compute Distances to Borders



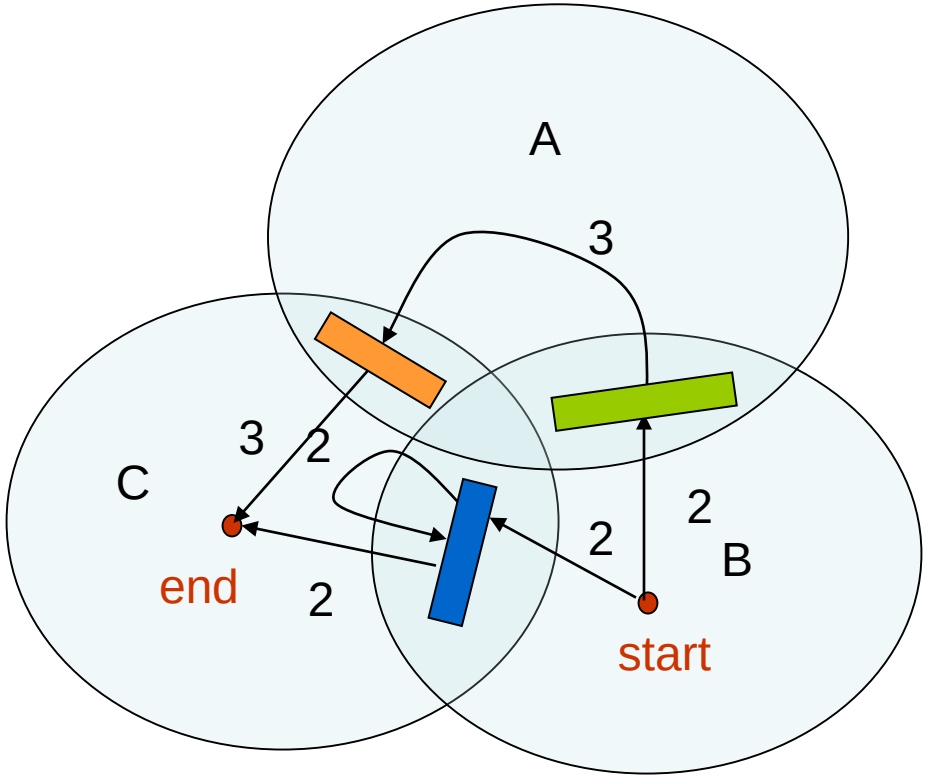
3) Add this Information to QPG

4) Find all paths from *start* to *end* (including cycles) <= k (10)

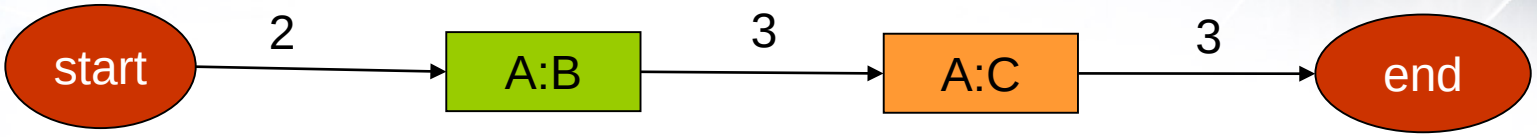
In this case 22 paths

5) Convert Set of Paths to Set of Queries

start – 2 → Peer_B:Peer_C – 3 → Peer_A:Peer_C – 3 → end



Converting Paths to Queries



- Each edge (pair of endpoints) represents a query
- For example, *ρ-path* (*start*, *Peer_A:Peer_B*, 2)

What is the correct hop-limit?

hop-limit = edge weight + (k – path weight)

ρ-path (*start*, *Peer_A:Peer_B*, 4) k = 10
ρ-path (*Peer_A:Peer_B*, *Peer_A:Peer_C*, 5)
ρ-path (*Peer_A:Peer_C*, *end*, 5)

Find the maximum hop-limit for each pair of end points

Pair	Hop-limit
(*start*, *Peer_A:Peer_B*)	5
(*start*, *Peer_A:Peer_B:Peer_C*)	7
(*start*, *Peer_B:Peer_C*)	8
(*Peer_A:Peer_B*, *Peer_A:Peer_C*)	5
(*Peer_A:Peer_B*, *Peer_A:Peer_B:Peer_C*)	5
(*Peer_A:Peer_B*, *Peer_A:Peer_B*)	3
(*Peer_A:Peer_B*, *Peer_B:Peer_C*)	6
(*Peer_A:Peer_C*, *Peer_A:Peer_B:Peer_C*)	3
(*Peer_A:Peer_C*, *Peer_B:Peer_C*)	6
(*Peer_A:Peer_C*, *end*)	5
(*Peer_B:Peer_C*, *end*)	8
(*Peer_B:Peer_C*, *Peer_B:Peer_C*)	6
(*Peer_B:Peer_C*, *Peer_A:Peer_B:Peer_C*)	5
(*Peer_A:Peer_B:Peer_C*, *end*)	6

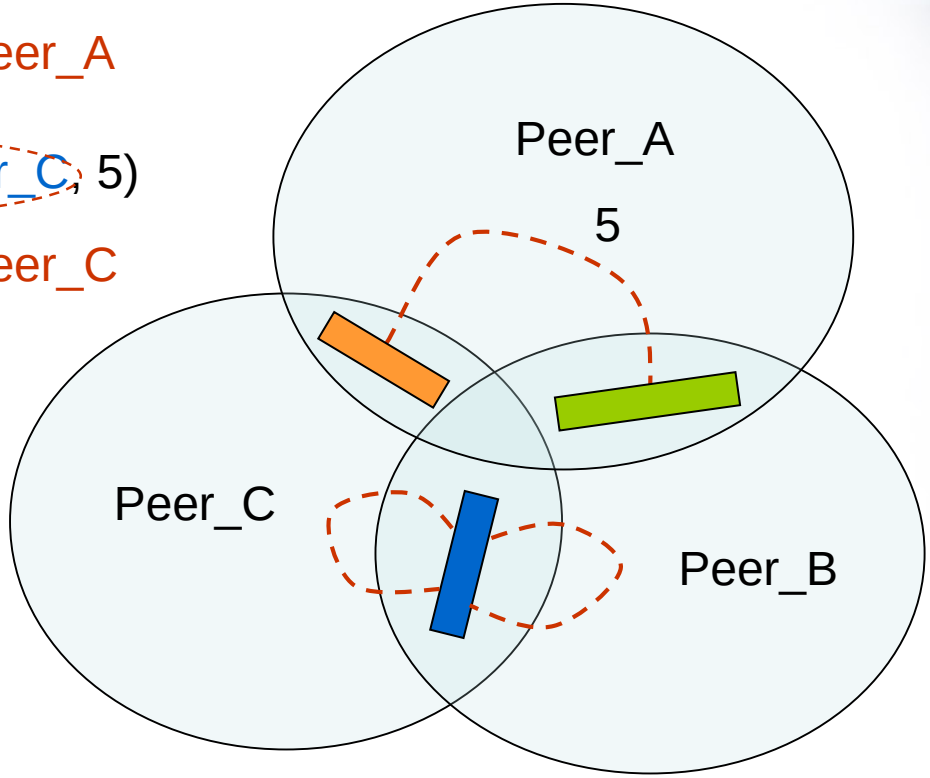
Which Peer gets each query?

ρ-path (Peer_B:Peer_A, Peer_A:Peer_C, 5)

Peer_A

ρ-path (Peer_B:Peer_C, Peer_B:Peer_C, 5)

Peer_B and Peer_C



Final Query Plan

Queries for Peer_B

FROM: Peer_B:Peer_C	TO: Peer_A:Peer_C	Hop Limit: 6
FROM: Peer_B:Peer_C	TO: Peer_A:Peer_C	Hop Limit: 5
FROM: Peer_A:Peer_B	TO: Peer_A:Peer_B:Peer_C	Hop Limit: 5
FROM: Peer_A:Peer_B:Peer_C	TO: Peer_A:Peer_B:Peer_C	Hop Limit: 5
FROM: Peer_A:Peer_B:Peer_C	TO: Peer_A:Peer_B	Hop Limit: 3
FROM: Peer_A:Peer_B:Peer_C	TO: end	Hop Limit: 5
FROM: Peer_A:Peer_B:Peer_C	TO: Peer_B:Peer_C	Hop Limit: 6
FROM: Peer_A:Peer_B:Peer_C	TO: start	Hop Limit: 7

Query Execution at Peer

Input:

Set of Queries: { ρ-path ({uri, …}, {uri, …}, k), …}

Algorithm:

Graph Traversal of Main Memory representation
Bi-directional BFS
Results in a set of statements

Output:

Union of each set of statements

Query Execution at Peer

- Peer does not enumerate paths
- Returns a subgraph (set of triples)

- Benefits
  - Eliminates redundant data transfer
  - Saves computation time

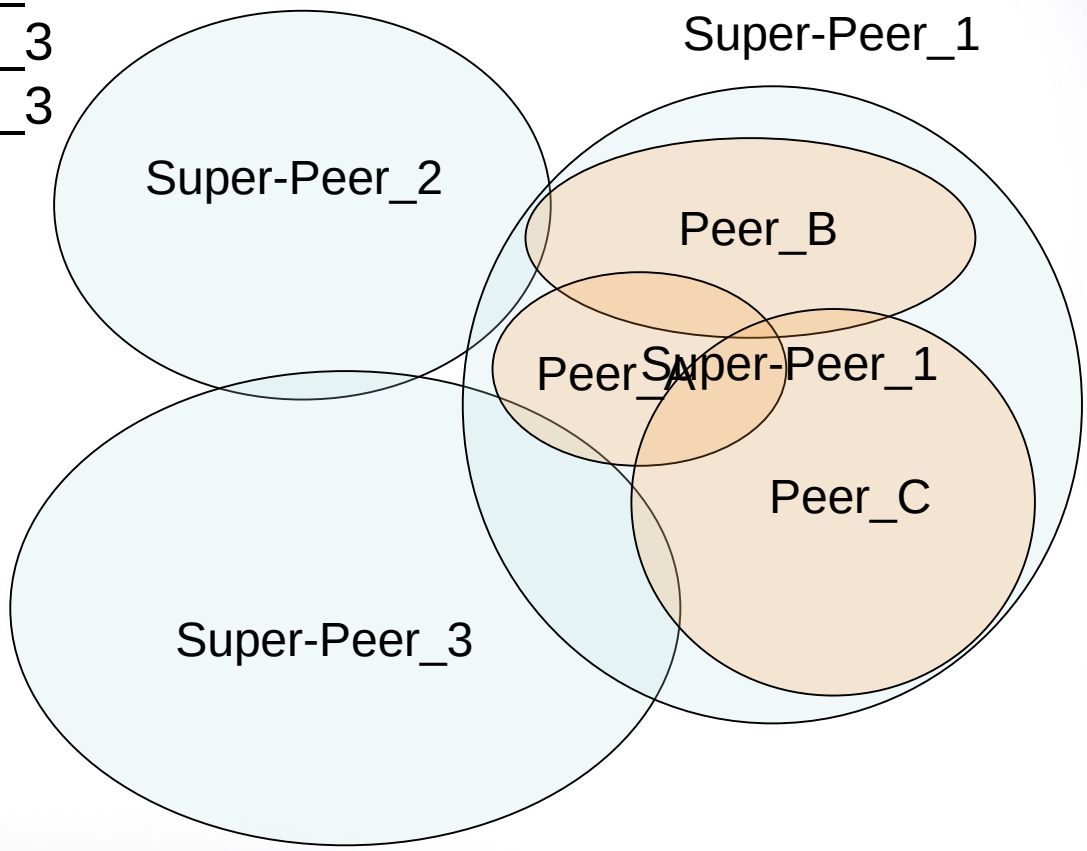
Scalability: Multiple Super-Peers

Super-Peer/Super-Peer Borders

- Super-Peer_1:Super-Peer_2
- Super-Peer_1:Super-Peer_3
- Super-Peer_2:Super-Peer_3

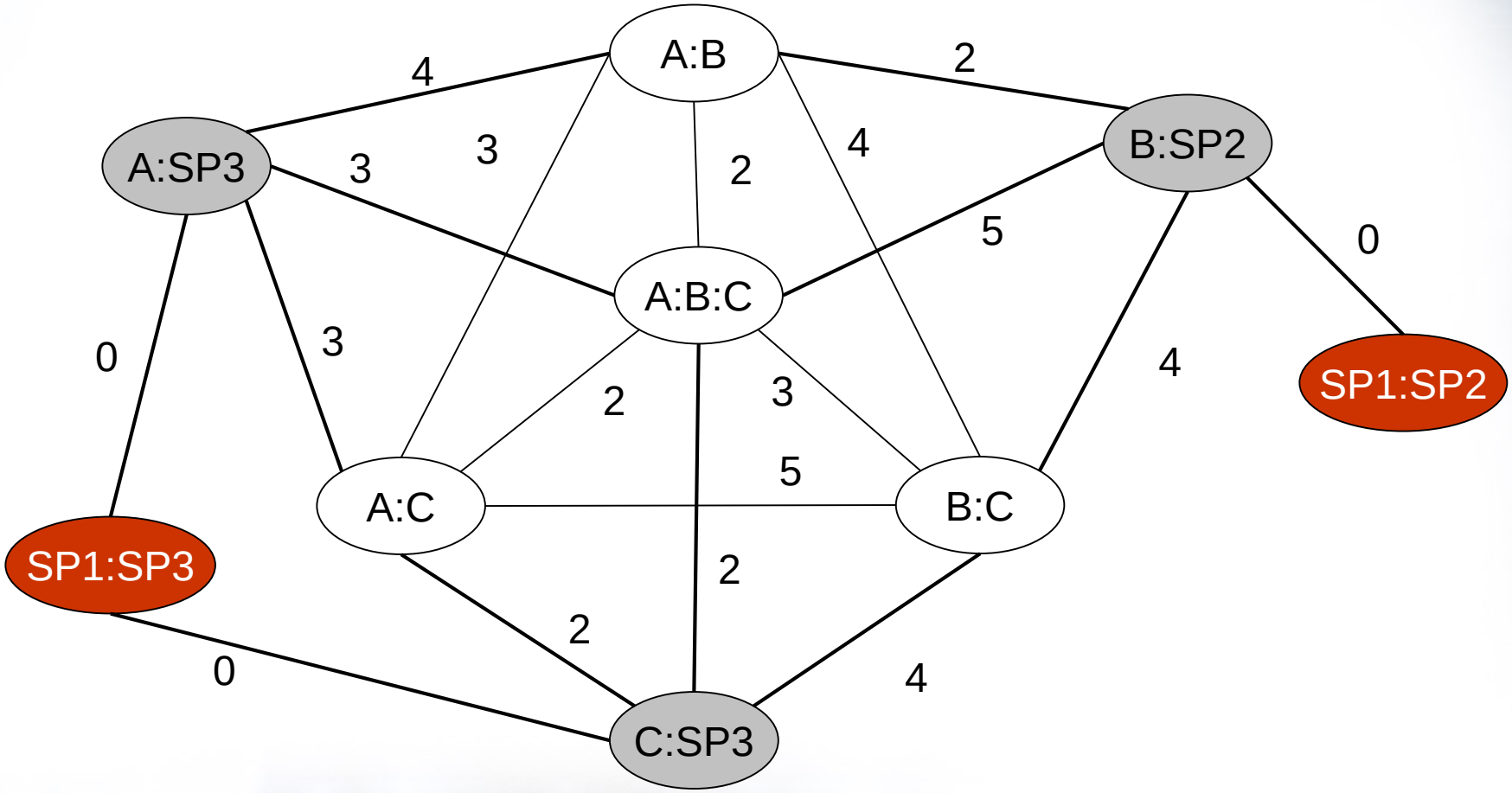
Super-Peer/Peer Borders

- Peer_B:Super-Peer_2
- Peer_A:Super-Peer_3
- Peer_C:Super-Peer_3

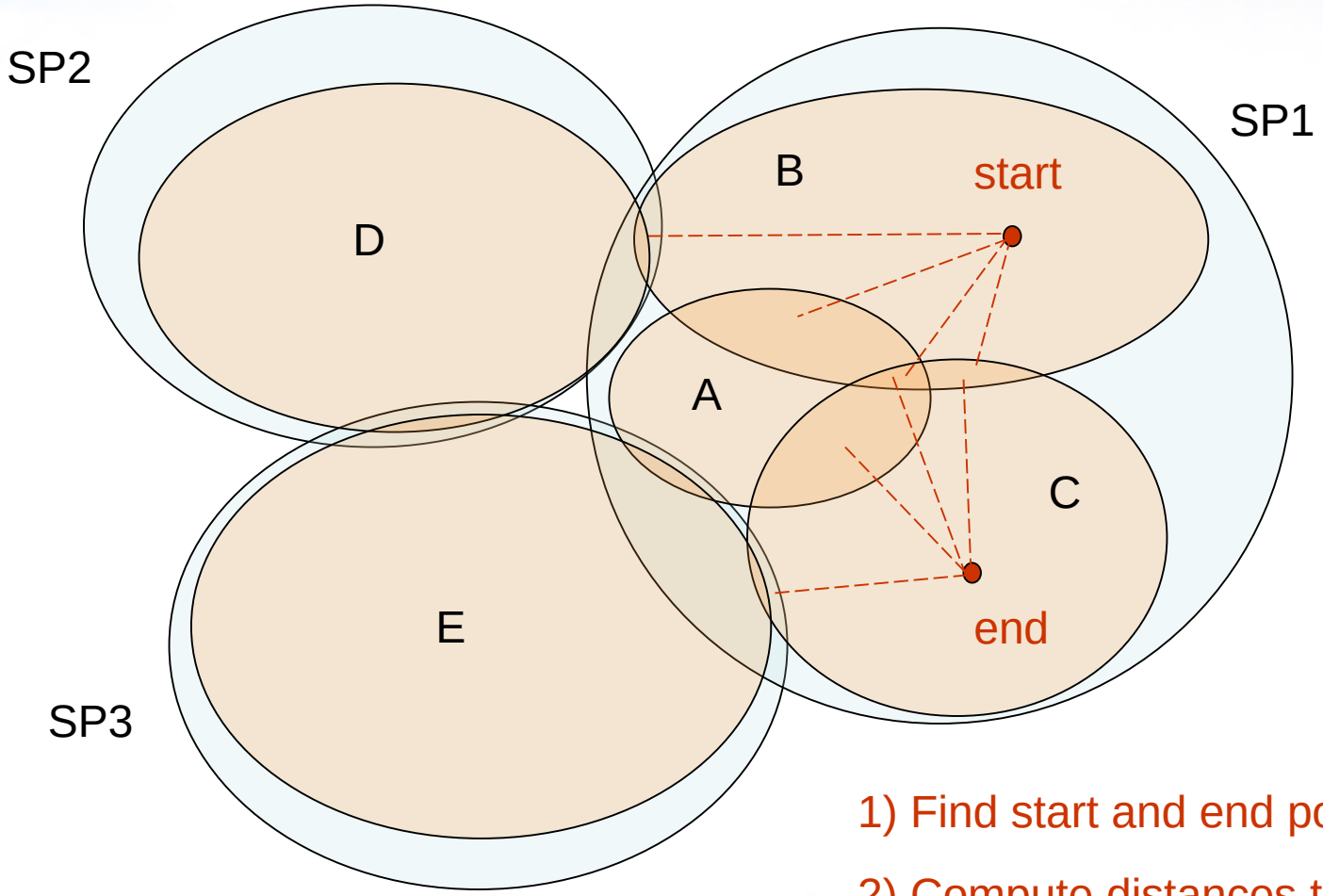


Integration of SP graph and Peer Graph

Super-Peer_1's new Peer-Level QPG



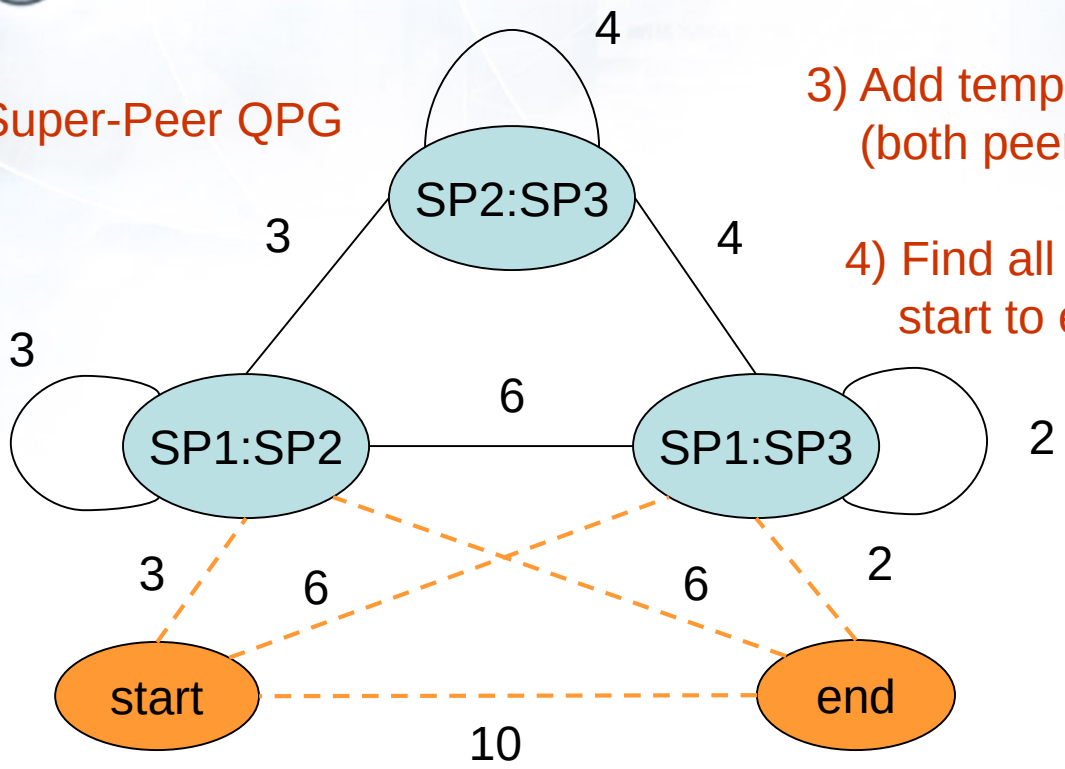
Query Planning Algorithm



1) Find start and end points

2) Compute distances to borders

Super-Peer QPG



3) Add temporary information for endpoints (both peer and super-peer QPG)

4) Find all directed paths <= k connecting start to end in the Super-Peer QPG

k = 10

start – 6 → SP1/SP3 – 2 → SP1/SP3 – 2 → end
start – 6 → SP1/SP3 – 2 → end
start – 3 → SP1/SP2 – 6 → end
start – 10 → end

5) Form a list of sub-query-plan requests for each super-peer

Super-Peer_1

FROM: start	TO: end	Hop-Limit: 10
FROM: start	TO: Super-Peer_1:Super-Peer_2	Hop-Limit: 4
FROM: SuperPeer_1:Super-Peer_2	TO: end	Hop-Limit: 7
FROM: start	TO: Super-Peer_1:Super-Peer_3	Hop-Limit: 8
FROM: Super-Peer_1:Super-Peer_3	TO: Super-Peer_1:Super-Peer_3	Hop-Limit: 2
FROM: Super-Peer_1:Super-Peer_3	TO: end	Hop-Limit: 4

Super-Peer_3

FROM: Super-Peer_1:Super-Peer_3	TO: Super-Peer_1:Super-Peer_3	Hop-Limit: 2

7) Each super-peer goes through the previous process on its peer QPG to form a list of ρ-path queries for its peers

Queries for Peer A:

FROM:	TO:	Hop Limit:
A:B	B:C	5
A:B:C	A:SP3	4
A:B	A:SP3	6
A:B	A:C	3
A:B	A:B:C	5
B:C	A:C	3
A:C	A:SP3	3
A:C	B:C	5
A:B:C	B:C	5
A:B:C	A:SP3	4
A:SP3	B:C	5

8) Querying peer now communicates directly with other peers to execute the ρ-path queries

Conclusions and Future Work

- Presented a Query-Planning Algorithm for r-path queries over distributed data set

- Problems
  - Efficiently compute node neighborhoods
  - How to continue searches across KBs
  - How to check for the many possible cases
  - How to determine search length in each KB

Conclusions and Future Work

- Future Work
  - Performance Testing
  - Effect of relative border size
  - Different criteria for group formation
  - How to accommodate other types of queries

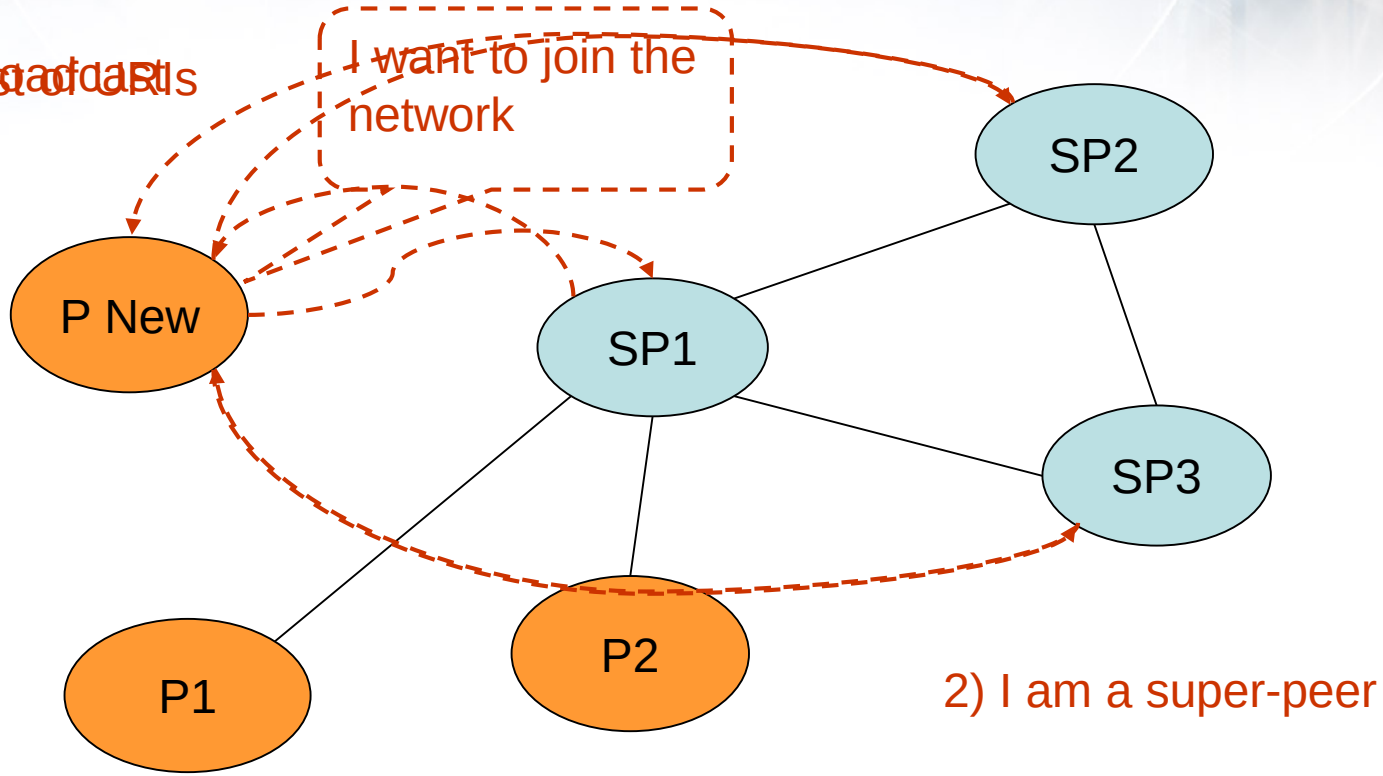
Questions?

Computing Borders

Super-Peer maintains Sorted Map of URIs

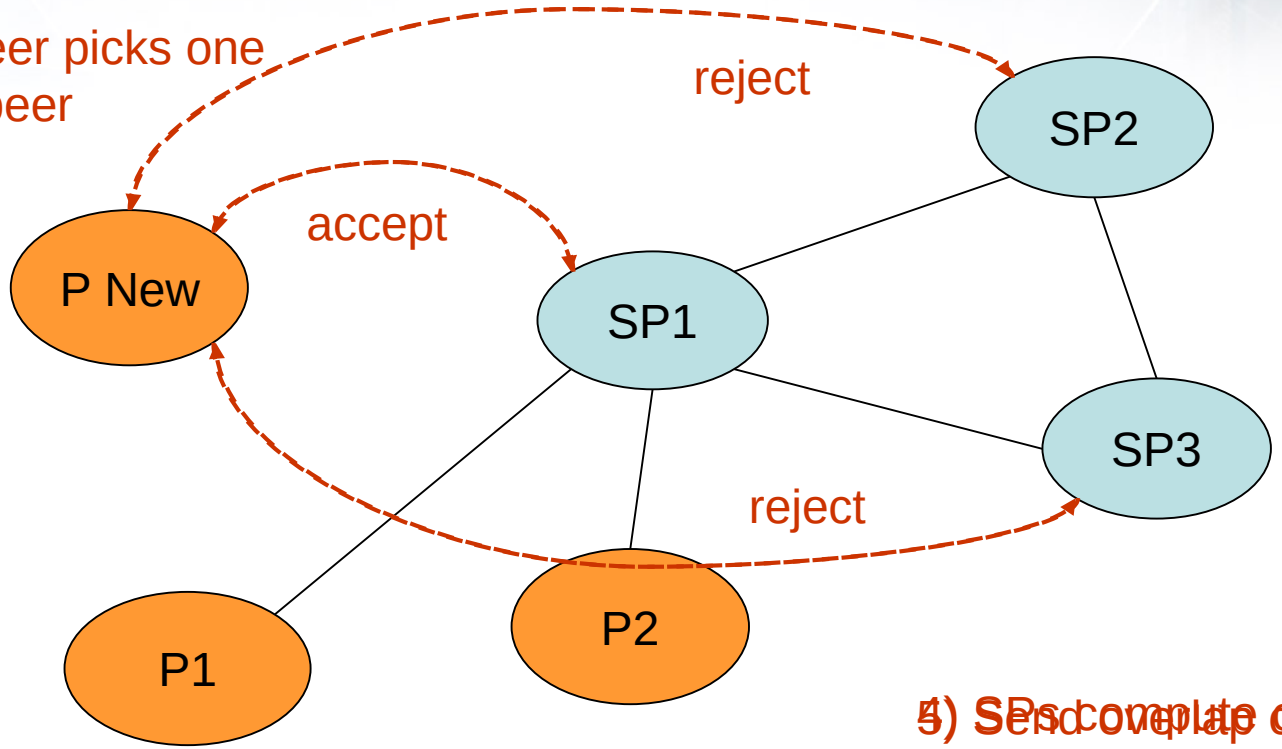
- Peer Border
  - Traverse new list and update Sorted Map
- Super Peer Border
  - Don’t care about other URIs not in this group
  - Keep total data transferred at a minimum

Forming the Network



Forming the Network

6) New peer picks one super-peer



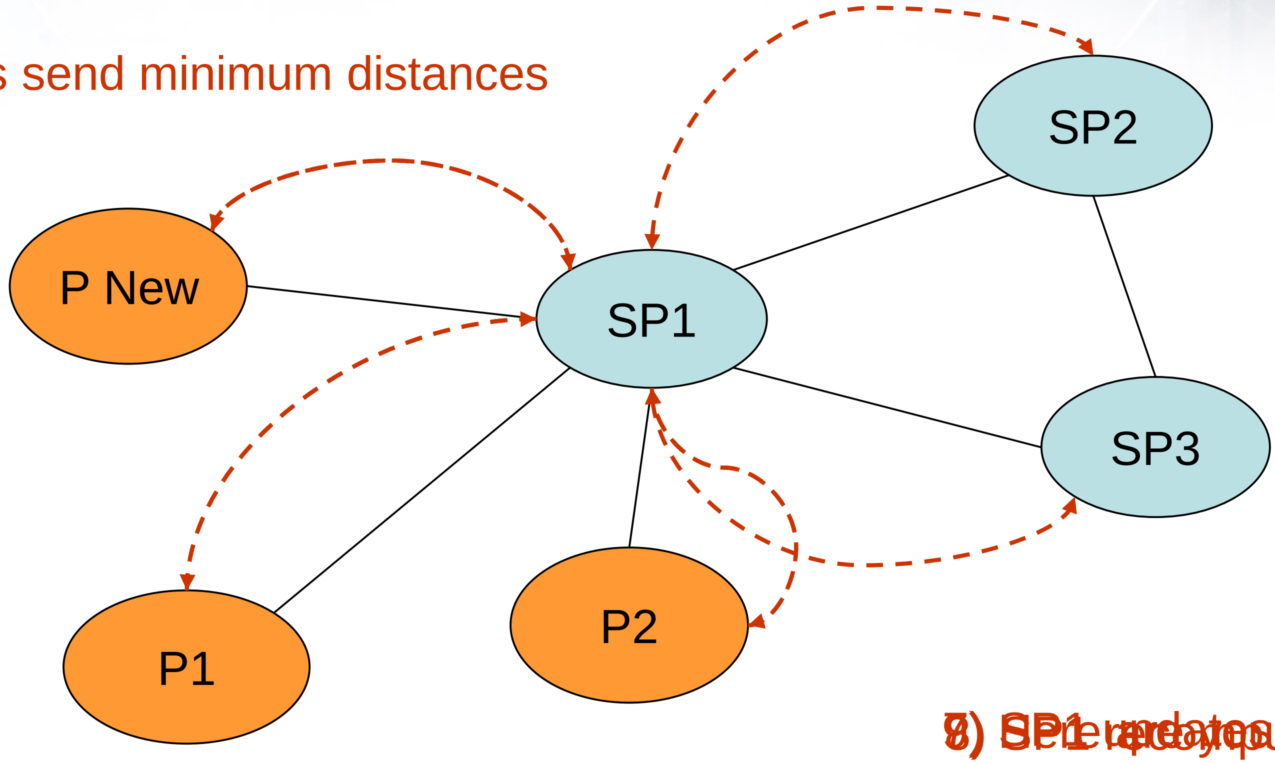
4) SPs compute overlap

5) Send overlap count to new peer

6) (maintain border information)

Forming the Network

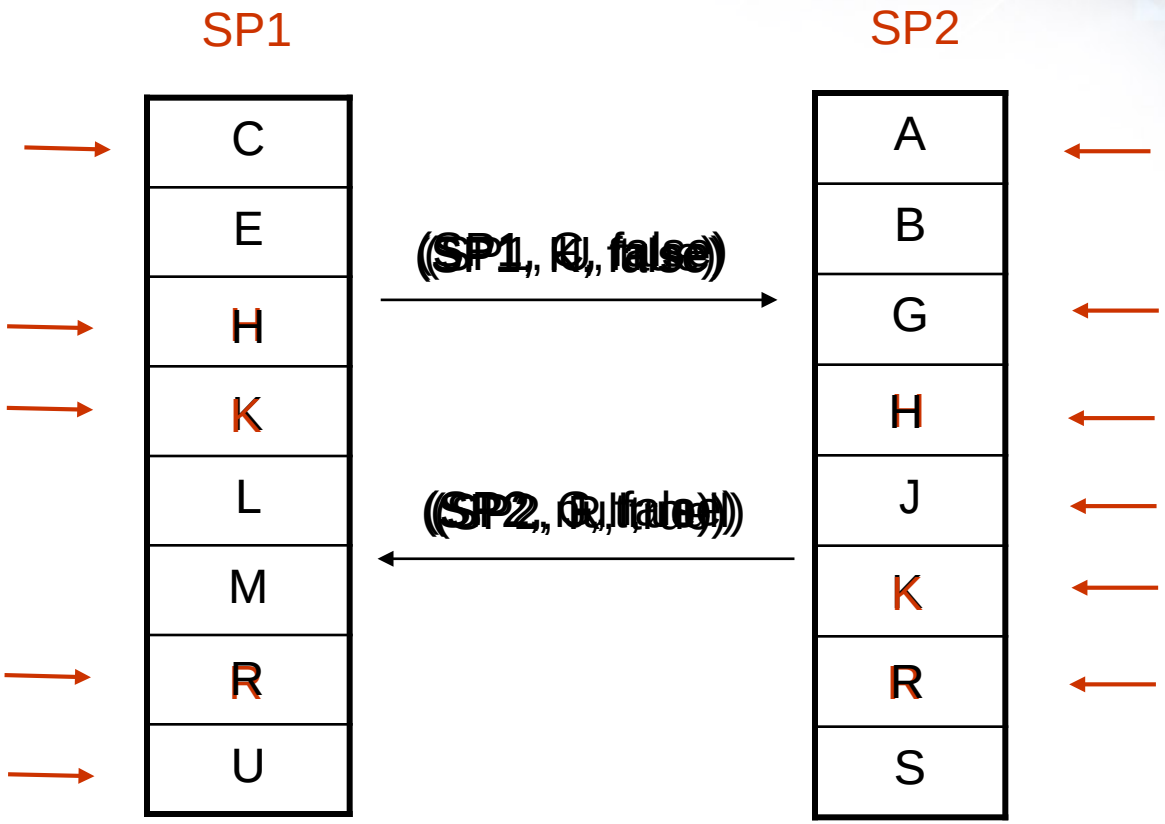
10) Peers send minimum distances



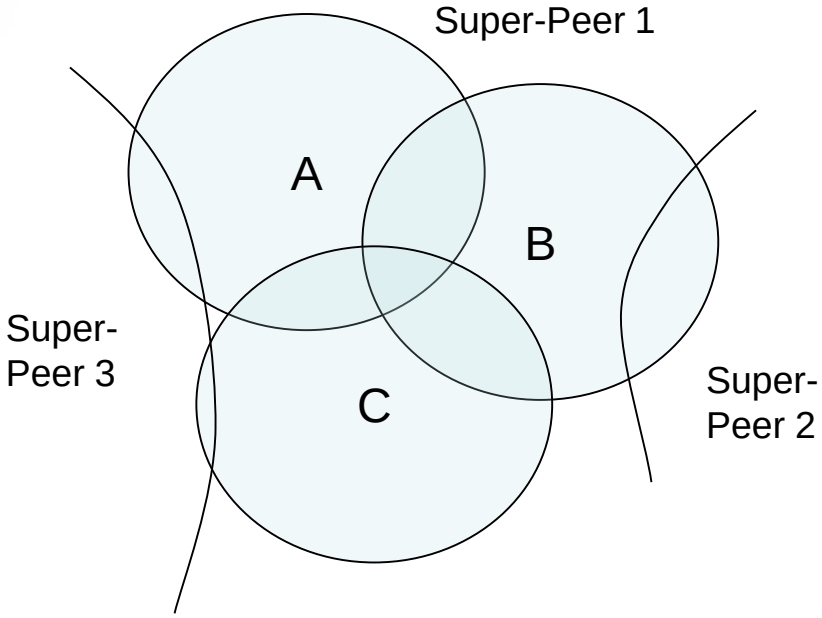
7) SP1 updates index

8) SP1 computes SP borders

Computing Super-Peer Borders



Super-Peer Level QPG



Borders
AB
AC
BC
A/SP3
B/SP2
C/SP3

Minimum Distances
dist (AB, BC) = 4
dist (AB, AC) = 3
dist (AB, ABC) = 2
dist (BC, AC) = 5
dist (BC, ABC) = 3
dist (AC, ABC) = 2
dist (AC, A/SP3) = 3
dist (AB, A/SP3) = 4
dist (ABC, A/SP3) = 3
dist (AC, C/SP3) = 2
dist (BC, C/SP3) = 4
dist (ABC, C/SP3) = 2
dist (AB, B/SP2) = 2
dist (BC, B/SP2) = 2
dist (ABC, B/SP2) = 2